Table of Contents

[Abstract 2](#_Toc174316932)

[Acknowledgements 3](#_Toc174316933)

[Chapter 1: Introduction 4](#_Toc174316934)

[I. Overview 4](#_Toc174316935)

[II. Research Questions 4](#_Toc174316936)

[III. Objectives 4](#_Toc174316937)

[Chapter 2: Background 5](#_Toc174316938)

[I. Dataset 5](#_Toc174316939)

[II. Literature Review 5](#_Toc174316940)

[III. Algorithms 10](#_Toc174316941)

[a. Neural Networks 10](#_Toc174316942)

[b. CNN 12](#_Toc174316943)

[Chapter 3: Methodology 15](#_Toc174316944)

[I. Tools and Techniques 15](#_Toc174316945)

[II. EDA and Visualization 15](#_Toc174316946)

[III. Project Plan 17](#_Toc174316947)

[Chapter 4: Results and Conclusion 19](#_Toc174316948)

[I. Results 19](#_Toc174316949)

[II. Comparative Analysis 20](#_Toc174316950)

[III. Conclusion 21](#_Toc174316951)

[IV. Future Work 21](#_Toc174316952)

[Chapter 5: Legal, Ethical and Professional Issues 23](#_Toc174316953)

[References 24](#_Toc174316954)

[Appendices 26](#_Toc174316955)

# Abstract

This attempt focuses on how CNNs can be employed in the context of cricket shots, using concepts in deep learning and computer vision. The main goal is to create a technique of identifying varied kinds of shots in cricket including legglance-flick, pullshot, sweep, and drive with the help of image analysis. In the shot classification, three different CNN models were built and experimented with; The success of the implemented models are analyzed with the metrics; accuracy, precision, recall, and F1-score. All the models were trained on a dataset containing a variety of cricket shots Strike prediction network was a CNN, the second level network was a deeper CNN network that has more layers and uses dropout regularization.

It can be seen that the simple CNN model offered a very good starting point but was not very efficient in learning more of the finer details of the cricket shots. The second model which we have named as model2, which has a greater number of convolutional layers than model1 along with dropout layer, gave the highest accuracy 91%. It proved to be the most reliable in the number of aspects it embraced while addressing the problem: it was a fine blend between the complex and generalized. However, the third model, which added more depth in the network, a little declined the performance which indicates that overcomplexity hinders the model’s ability to be trained and perform.

It also comprehensively explains the related legal, ethical and professional concerns in the employment of image data for sports analytical purposes including ownership of image data, Consent of the expected image data subjects, particularly the players and the likely shift of roles of a coach or trainer. While this work has its significance in providing a solution to several inferences in sports analytics, it also provides a foundation for the next steps to be taken in this research area, such as temporal information and multimodal data analysis, as well as real-time analysis. The results also point at the necessity of the proper choice of the model architecture depending on the intended applied methods and necessary performance-to-complexity ratio in the case of the method applied in this paper to the cricket shot classification.

# Acknowledgements

# Chapter 1: Introduction

## Overview

Cricket is a sport that is famous worldwide with multiple countries participating in it every year. There are multiple variations of shots that a player can play in each turn. By using the power of deep learning and computer vision (*Computer vision in sports: applications and challenges*, no date), a deep learning system can be trained to classify the kind of shot a player played. The dataset available on [Kaggle](https://www.kaggle.com/datasets/aneesh10/cricket-shot-dataset) will be augmented (Sharma, 2021) to increase the number of records. To conduct this research two frameworks PyTorch (Simplilearn, 2021) and TensorFlow (Banoula, 2020) will be used and to construct and train the models.

A main challenge in this research is each player has his own unique posture and style of playing a certain shot. Existing methods for this classification is manual where a person manually looks at the shot and labels it based on the domain knowledge. This research project aims to advance the use of AI in sports. By comparing two different frameworks, more insights can be gained. Though this project only focuses on trying to predict the types of cricket shots from images, in further research it has the potential to do this using a video input or even live streaming. The hardware and software requirements may be increased but there is high scope of research in this area.

Based on the development of the prior arts of sports analytics, this research aims at incorporating the function of deep learning to define cricket shots since it usually requires professional human intervention. While the results of the manual classification are quite accurate, the procedure is tedious and the outcome depends on the researcher’s subjectivity. In this study, the CNN is used and the objective is to develop a system that is very efficient in recognising and classifying cricket shots to reduce on biased analysis and time taken during analysis. Besides the comparison of PyTorch and TensorFlow frameworks, it is needed not only to find out which one is more effective for this task but also to help to develop the discussion of the best frameworks for the realization of deep learning in computer vision tasks.

The project also seeks to identify the issues and challenges when using image data in the analysis of sports. Because of the changes in player postures, environmental conditions and orientation, and camera views, the classification task is challenging, meaning that more than basic data preprocessing and data augmentation techniques can be used to enhance the model’s performance. Although this work is more on the side of image-based classification, it creates the foundation for extended real-time shot analysis that would in the near future totally bring about a change in the trend of coaching and approach aspect of cricket. The findings derived from this work are compendious and may be useful for works in and outside cricket especially where motion and action classification are vital for performance evaluation and strategizing.

## Research Questions

Performance of Pytorch deep learning algorithms in terms of training time, accuracy, precision, and F1 score when classifying different cricket shots from image data?

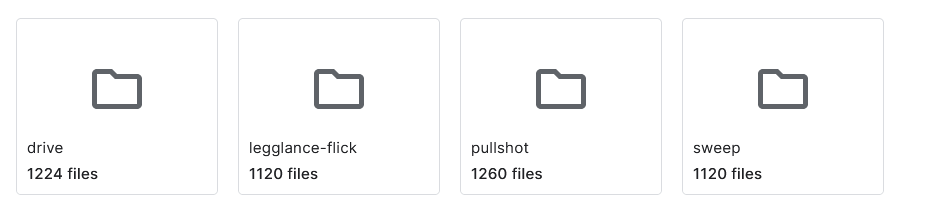
## Objectives

* To clean and preprocess the cricket shots dataset for training a DL model.
* To implement model a deep learning model development using PyTorch.
* To compare multiple models based on multiple metrics like speed, accuracy, precision and f1 score.
* To evaluate the performance of this system on real life data.

# Chapter 2: Background

## Dataset

The dataset contains four classes each of which represent a type of shot in Cricket. Each shot is a class in this dataset, for each class there are almost 1100 to 1200 images present as shown in the picture below. Together the total dataset has around 4400 to 4600 images. Most of the images are in “.png” file format which is an image file format. All the images have 3 channels – Red, Green and Blue. The images consist of a diverse range of players practicing in nets to playing in the world cups, there are players of multiple nations with varying jerseys.



Ex images:

## Literature Review

Paper I:

The paper by (Bhat *et al.*, 2023) aims at collecting a clean dataset for identifying cricket shots as similar datasets are considerably rare and most of them are not publicly available. This is done manually where cricket shots from IPL 2022 videos are named utilizing VGG software and then the FFMPEG software is used to create video clips. The dataset comprises 1922 clips across six shot types: butter-cut, drive, flick, pull, slog, and sweep. The PES dataset is then compared to a reference set by Gupta et al as well as the UCF101, a transfer learning based off a pre-trained ResNet-50 model.

Methodology

The method employed for the extraction of the measure revolves around the raw data and the model inspection. The process of video annotation carries out in the manual annotation stage is mainly the labeling of video segments with shot types and the check of clip quality. It is very manual and time consuming and should be validated by an expert to ensure the results obtained are correct. The subsequent generation of shot clips is done using the FFmpeg software where the annotated segments are organized into a structured dataset. Mwe PES which has clips of the resolution of 720p with 30 frames per second, is compared with two other datasets using transfer learning based on ResNet50, tested across three different ratios of train-test (70:30, 75:25, 80:20).

Metrics and Values

These are some of the commonly used KPIs; accuracy, precision, recall, F1 score. It also depicts that the proposed PES dataset performs better than the reference dataset with an average accuracy of 50% against to 38%. 33% for the latter. In the 70:30 train-test split applied on the PES dataset, the related precision is equal to 0. 58, recall of 0. 70; precision of 0. 83; recall of 0. 87; and F1 score of 0. 64, with the reference data set listing 0. 17, 0. 10, and 0. 13, respectively. While keeping a precision of 0. for the 75:25 split, the PES dataset has a stable accuracy. 67 while the DA of the reference dataset was 0. 50. In the case of the 80:20 split, the PES dataset receives 0. 67 on the measure of precision and 0. 20 for recall, and an F1 score of 0. number is 31, while in the reference dataset it is 0. 50, 0. 10, and 0. 17.

Pros and Cons

Firstly, the specific research has presented a number of advantages; creating a high-quality vast video dataset for classify cricket shot, which is very useful for a real-world application and fulfills the critical limitation of availability of data for sports video processing. That is why, the PES dataset reveals higher performance in terms of multiple parameters in comparison with others, which proves usefulness and adequacy of the suggested dataset.

But one must bear in mind that the very construction of the methodology is based on manual annotations, which can be very time consuming and may not fit large projects. Anyway, the process is not fully automated, and this can become a limitation for the general use and dissemination of the dataset. The following may be targeted for improvement in future; The process of annotation: with the help of advanced techniques, it should be automated, the data: the dataset should be expanded to include multi-modal data that should be used to develop few-shot learning models.

Paper II:

The research study by (Fernandes *et al.*, 2023) responds to the emerging demand for employing the CV and DL approaches for automatic identification and categorization of cricket shots from images. Cricket being one of the most popular sports across the globe has also experienced rising trends in the use of technology in the performance analysis and coaching. The study argues on designing a system that involves the utilization of a 2D CNN with more layers such as convolution, pooling, flattening, and fully connected layers. The accuracy of the model is 91 percent thus the model is accurate. It was able to achieve 5% in detecting various types of cricket shots which can be used in the real world.

Methodology

The approach entails the creation of a 2D CNN architecture with the intended purpose of categorizing cricket shots from images. The model’s architecture includes convolutional layers having filters of size 3 \* 3, each of which was 32; ReLU activation functions are used. After the second convolutional layer of the network, a pooling layer with 2 x 2 window is included to decrease field. The output of the pooling layer is then flattened this is fed into a fully connected layer with 128 nodes. Last, the model applies SoftMax activation function to the output layer in order to filter the input image to one of the six cricket shot. All of this is conducted using Keras, with TensorFlow as the backend, and the model is trained using the ImageDataGenerator class for augmentation.

Data pre-processing is an important step of the whole process, which includes conversion of video frames into images, resizing of images and using filters for improving the quality of images. To authorize an accurate evaluation of the developed model, the database is divided into input train, input valid, and input test. For the training, backpropagation technique is used to minimize the loss function with validation steps to make the appropriate adjustment of the model and avoid overfitting. The model is first applied to the test set; the obtained results can be used to determine accuracy, precision, recall, F1 rating.

Metrics and Values

It reveals a total accuracy of 91 percent and confuses the letters of different case more often than the numbers. 5% for another model incorporated in the research known as 2D CNN with the aim of identifying and categorizing cricket shots. This measure reflects the share of the cases when the model gives the right prediction of the result out of all its predictions. Results similar to those on the unknown data from a distinct test set further support the model’s validity. The SoftMax activation function in the output layer guarantees the production of probabilities for each class with the likelihood of the network’s choice given by the highest value.

Pros and Cons

The study has several strengths, and this is with regard to the use of deep learning to solve a practical problem that is related to sports analytics. Thanks to the achieved accuracy the model appears to be beneficial in a variety of tasks, starting from the coaching systems and up to performance analysis and even live sports broadcasts. Thus, the application of a 2D CNN, which originates from image classification problems, creates a solid base for solving the task, while applying data augmentation contributes to improving the model’s generalization.

However, it is also important to state certain limitations of the study. The dataset into which the model is trained and tested has few images 180, so there is limited variation of the cricket shots that may occur in different match situations. Additional work might have to be done in future where the model’s performance could need to be tested again based on a more comprehensive database of a more diverse nature. Also, the proposed study examines only the imaging analysis of the cricket’s shot, and incorporating additional data from other sources, for instance, video or sensor data, can be beneficial and give a deeper insight into the crickets’ shot tendencies. This also applies to the reliance on manual data collection and preprocessing which hinders scalability and thus the use of automated methods to enhanced the current approaches.

Another disadvantage is that much consideration is given to other deep learning architectures that might be more accurate or that take less time to perform inference. The study could benefit from a comparison of other models like 3D CNNs or recurrent neural networks (RNNs) that have been discussed earlier because they are suitable for temporal data especially videos. Additionally, it could have examined the effect of hyperparameter tuning on the models’ outcome, as tweaking the parameters such as learning rate, batch size, and layer number could enhance the results.

Thus, the applied investigation proves the applicability of 2D CNNs to identify the cricket shot from images with fair accuracy rates. Yet, there seems to be space for further enhancements in data-set heterogeneity, model evaluation, and data preprocessing scripts’ integration. Further research could be done to extend the cited areas; bringing improvements to the applicability and scalability of the proposed system would be beneficial for the field of technology in sports analysis.

Paper III:

The paper under review (Jagadeesh, Rithesh and Sagar, 2023) tries to identify and survey the most relevant methods for deep learning and machine learning for cricket shot detection and classification. Since cricket has gained more popularity than before and due to the progress in the field of hardware and computational power, computer vision and deep learning on cricket match analysis has gained more emphasis. The survey is primarily aimed at the selection of the most appropriate algorithm for distinguishing cricket shots based on the images by employing CNN and transfer learning as the major approaches.

The primary focus of the survey is to discover and discuss CNNs as well as transfer learning, which are considered to be the most successful methodologies for the given problem of cricket shot classification. CNNs are famous because they can learn the features or patterns from the images without any human intervention of feature engineering. The paper also states that CNNs are ideal for cricket shot recognition because of their ability to analyze and distinguish the various visual characteristics often involved in different shot types, such as the position of the bat and the players. Also, transfer learning is presented as the effective approach that uses the pre-trained models developed with the help of the big data, including VGG-16 and AlexNet, to increase the accuracy of models, if they work with the small data. This means that it is possible to fine-tune the pre-trained models like the Imagenet that has features which can be tuned to recognize features important in discriminating various cricket shots instead of starting the training process from scratch, hence cutting down on computational power use and time.

This paper also presents the applicability of other machine learning algorithms such as LSTM networks, CDBN, ISA, SVM, and OCR. LSTM networks are particularly useful when working with sequential data and, given the fact that cricket videos involve time series, this is definitely an advantage. CDBN and ISA are introduced for their feature of establishing complex spatial and temporal relations of the data and their disadvantage of large requirement for annotated data and their susceptibility to the risk of over-training. Some of the features that have made SVMs famous include its ability to cope with noisy data, a big downside to SVMs is that they are somehow sensitive to the choice of their parameters and do not perform well when given imbalanced data. In reference to the cricket footage, OCR is emphasized for its capability of extracting written information from the TV screen; however, its usefulness is restricted to circumstances where on-screen text or graphics are available.

The paper can thus be considered to distinguish between the strengths and drawbacks of each technique as well. CNNs and transfer learning are also considered accurate and fast in their ability to perform the role of image classification. The two possibly advantageous areas of its application are noted, although the use of pre-trained models in transfer learning is the most effective, as it makes it possible to distinguish between models with high accuracy using comparatively small datasets. However, it seems that tying into large pre-trained models can decrease the utility of transfer learning in applications where the data set is highly specific or where there are many categories of cricket shots.

LSTM networks, despite having been proved to be efficient in the modeling of sequential data, are computationally intensive and receive poor training from small, well-annotated datasets. CDBN and ISA, on this account, are relatively powerful but intricate and computationally intensive models and may, therefore, not be easily applicable on most problems. SVMs, though quite resistant, sometimes encounter issues with handling imbalanced classes while OCR occurs in certain situations only if the information is textual.

Thus, the paper offers a comprehensive review of the recent advancements in the identification and categorization of cricket shots accompanied by their classification in terms of CNN and transfer learning. However, the survey also intensively pinpoints the direction for more research on the applicability of these techniques over the cricket domain and issues regarding massive scale data labeling and the interpretability of such models. The information collected in this survey will be made useful for subsequent work in the process of identifying and creating new and more accurate and labor-effective methods for the description of cricket shots.

Paper IV:

The research paper (Mannan *et al.*, 2021) under review addresses the problem of classifying cricketing activities through deep learning approaches, which include CNN and transfer learning. To this end, the study presents a new dataset known as the cricket image classification (CIC) dataset, which aims to depict several incidents concerning cricket such as various cricket strokes and the umpire’s signals. The researchers use not only ResNet50 architecture but also others like ResNet34, ResNet101, VGG16 with Batch Normalization, and AlexNet to check which of them performs better in terms of the accuracies of the classification of these activities. Thus, by applying the One Cycle Policy for managing the training process, the research reveals the accuracy of 99%. Hence, the success rate of their strategy stands at 47%, clearly showing high efficiency.

Methodology

The methodological approach of this study is founded on CNNs with special focus on ResNet50 considering that the model has been reported to have high generalizability in image classification tasks. The researchers adopt transfer learning; therefore, the study begins with a pre-trained ResNet50 model that was originally trained on the ImageNet database. This approach can dramatically cut the time it takes to train the model and the amount of computational power needed while keeping the same amount of accuracy. The study also uses the ADAM optimizer alongside one Cycle policy to further improve the training process’s efficiency. The One Cycle Policy entails the provision of rules for the alteration of the learning rate which is useful for over-fitting avoidance and increased rate of convergence.

The dataset used is CIC which includes 3,233 images belonging to 9 different classes of cricketing activities. These images are captured from the professional cricket grounds and also from the street cricket, this is the way to have the large number of pictures with the difference settings, so that the dataset will be very diversified. The images are preprocessed and passed through a channel ordering conversion, resizing, one hot encoding of labels and augmentation to fit the models’ training.

Model Selection and Performance

Different capabilities of CNN architecture are described in the study to select an ideal CNN for the cricket activity classification. ResNet50 is selected as the primary model because of its accuracy and efficiency given that it has less layers as compared to other deeper networks. The researchers also check ResNet34, ResNet101, VGG16 with Batch Normalization, AlexNet for comparison of performance of various architectures.) For each of the models, the last layer is modified to target the number of classes defined in the CIC dataset, instead of the layers originally developed to work with the ImageNet dataset.

Indeed, by applying the ResNet50 model with the One Cycle Policy, you will reach the maximum level of accuracy, which is 99%. On the test set, it achieving an accuracy of 47% which is higher than the other models. The One Cycle Policy which involves a higher learning rate in the early phase of the training, and decreasing the value in the second phase reduces the number of epochs needed for training; ResNet50 gives better top performances in just 12 epochs. This is much better than general learning patterns where over 50 epochs are necessary to get the very same accuracy rates.

Pros and Cons

The main contribution of this work is a method for improving the process of training to increase accuracy and minimize computation time with the help of One Cycle Policy connected to the deep learning. The application of transfer learning is another substantial benefit because these models are trained on large portions of knowledge and, in the presented work, the CIC dataset is relatively small and heterogeneous.

In this paper, a novel method is introduced to identify the cricket activities with the help of deep learning system. We have trained our model on own dataset and own dataset contains images captured from different angles and are very much capable to detect and differentiate between incidents of cricket. Due to the absence of similar works, this work was extremely arduous; however, the suggested model resulted in an accuracy rate of 99%. 4 7%. Earlier works addressed this task in various form and manner, applying different methods to the task, yet, the outcome was not as satisfactory. In an absolute and unparalleled manner, we have transferred learning and augmented the dataset, optimization techniques, optimizer and other hyperparameters in a way that distorted our model so vividly to generate such a perfect accuracy rate and at the same time made the model more robust and time efficient.

## Algorithms

### Neural Networks

Neural networks belong to a group of machine learning models that follow the style and structure of the human brain. It is made of layers of interconnected nodes, or neurons, and each link between nodes has a weight. These networks are primarily used to identify patterns in the data, to be able to learn from the given data and then make predictions/classifications from what has been learned. It must also be said that neural networks are at the heart of deep learning, which allows for the development of most modern AI solutions ranging from image and speech recognition to natural language processing and autonomous systems (Sawtell-Rickson, 2022).

A neural network normally consists an input layer, one or more hidden layers, and an output layer. Each layer is neurons that are in fact mathematical functions taking the inputs from the previous layer, applying weights and biases, and then passing the results through an activation function before passing the result to the next layer. Introduction of activation function brings in the non-linearity which enables the network to learn higher level features of the data. During training, the weights and the Bias terms coefficients are changed to minimize the gap between the network output and the actual results that is called learning (Sarita, 2023).

**Hyperparameters in Neural Networks**

Hyperparameters of a neural network are the parameters that remain unidentified by the data and are established before the training session (Radhakrishnan, 2017). These parameters affect the performance and efficiency of the network in a tremendous way. Key hyperparameters include:

**Learning Rate:** It defines how big the steps during the training are that is how big a change the network makes to the weights. Training at a higher learning rate can train fast but the network overshoots the values and become suboptimal. The learning rate is the rate at which the algorithm changes on each epoch, a model with a small learning rate will make smaller adjustments but will take more time in training that is on more epochs.

**Batch Size**: This is the number of training samples processed before the network’s weights are adjusted for better performance. Bigger sizes of batches increase the stability of the training, and therefore the fluctuation is less but it slows down the convergence. Larger batch sizes offer the opportunity to get a more accurate average of the gradient, but at the same time give better update but using more memory.

**Number of Epochs:** An epoch is the number of cycles of all the data that is used for training before it is presented again for another cycle. Epochs represent the quantity of cycles of the learning process for going through the entire dataset. When the epochs are very few, there is underfitting in which the model does not study the data patterns very well. If the number of epochs is very high then the model tends to learn the noise also, which is known as overfitting.

**Number of Layers:** In the context of the current understanding, a depth of a neural network refers to the number of hidden layers. Greater numbers of layers may cover more intricate patterns; but there lie the dangers of over-learned and in addition the training process of the network becomes difficult.

**Number of Neurons per Layer:** This defines the number of neurons in a layer, thereby defining the capacity of the model at the layer. The advantage of more neurons is that the model includes more details in the target domain; however, this implies the issue of overfitting and computational cost.

**Activation Functions:** Activation functions bring non-linearity into the model so as to make it capable of learning in order to mimic certain patterns. Common activation functions include:

* Sigmoid: Rescales the output to the range of [0, 1] which can be utilized in the output layer when targeting a binary classification problem.
* Tanh: Scales the output to a value that are between -1 to 1, often applied in hidden layers.
* ReLU (Rectified Linear Unit): All the negative inputs are set to zero and it is easily implemented and quite effective hence making it popular in hidden layers.
* Leaky ReLU: Like ReLU it has positive derivative servicing the purpose of sparsity and the difference for negative values is small to avoid vanishing gradient.

**Dropout Rate:** They also include Dropout which is a form of weight decay that ‘drops out’ a fraction of the neurons and their connections during training to avoid overfitting on the training data. This minimizes over training and increases the accuracy of the generated model and makes it more generalized for use.

**Weight Initialization:** The type of initialization employed on the network’s weights may also affect the rate at which the training is performed or the efficiency of the training. There are other methods such as random initialization, Xavier initialization, and He initialization with specific relation to activation functions and appropriate networks.

**Optimizer:** It stops the learning process and decides the method by which the weights of the network are altered using the gradient of the loss function (Gupta, 2021). Popular optimizers include:

* **Stochastic Gradient Descent (SGD):** Adjusts weights based on a relatively smaller subset of training data which while noisy often proves to be more effective.
* **Adam (Adaptive Moment Estimation):** Adapts works with other two other stochastic gradient descent extension that retains exponentially decaying average of past gradients (M) and past squared gradients which is helpful for convergence.

**Loss functions in Neural Networks**

The loss function or known as cost function is a function that evaluates a prediction of a neural network against the actual values. The outcomes of the component are essential in determining the progression of training. This is the loss that training seeks to avoid (Yathish, 2022). Common loss functions include:

* **Mean Squared Error (MSE):** Also applied for regression problems, MSE calculates the mean of the squared differences between the forecasted values and the real ones. It is less robust; large errors therefore are even more unaccepted.
* **Cross-Entropy Loss:** Implicitly used for classification tasks particularly in the binary and multi-class classification problems. It quantifies the difference between the probability distribution predicted by the model and the actual probability distribution, which imposes penalization on the outliers from the real class.
* **Hinge Loss:** Some of them are applied for training classifiers possibly in the context of support vector machines (SVMs). It rewards correct predictions and punishes the wrong and/or the ones made with low levels of confidence.
* **Huber Loss:** Merging MSE and absolute error loss, Huber loss is less affected by the outliers compared to MSE but they are punished as well, making the function suitable for the regression tasks where robustness is required.

**Working of Neural Networks**

Neural networks allow the input data to be processed through the layers with weights, biases and activation functions which have their initial forms in the layers. Such a procedure is known as forward propagation (Simplilearn, 2020). The last calculated output is then subtracted from the target values via the help of a loss function to get the value of the error.

When performing training during the execution of the code, with this kind of error, rarely it is not controlled through an optimization algorithm like SGD or Adam, with the technique known as backpropagation. Through backpropagation, the gradient of the loss function with respect to each weight and bias of the network is computed so that the optimizer can alter it in a way to decrease the error in the subsequent predictions.

Every epoch, the network’s weights and biases are adjusted until it is able to guess most of the test data correctly. Thus, by the end of training, the weights allocated to the neural network must be such that when the net is faced with new data, it can generalize this data well; hence, the use of neural networks in various applications (Simplilearn, 2020).

### CNN

CNNs are categories of the neural networks used mainly for the image and video analysis. Unlike other types of neural networks that may have a problem with the high dimensionality of the image data, CNNs exploits the spatial structure of the image and can detect, for instance, edges, textures, and other features superior to simple edges. CNNs are a central of many modern computer vision applications such as image classification, object detection, and facial recognition (Chatterjee, 2019).

**Basic Structure of CNNs**

The basic parts of CNNs are layers that execute different computations: convolutional layers, pooling layers, and fully connected layers. Every one of these layers is significant for the work of a network as a whole (Sengupta, 2023).

**Convolutional Layer:** The convolutional layer is at the heart of a CNN and is possibly the most important layer of a CNN. It involves convolving an image with a filter or kernel to arrive at an output referred to as feature map. When implemented in a timely manner, this process is known as convolution. It moves across the input image and for each position dot-product is computed, which sums up the characteristics of the image space (Malviya, 2023).

**Filters/Kernels:** Filters are small matrices of weights which aims at identifying numerous features in the input. For instance, in the early layers we might have filters that distinguish edges of different types or the type of texture, while in the deeper layers we might have filters that distinguish between different faces, or different objects.

**Stride:** Hence, Stride refers to the manner or the pace that the filter moves through the input image. Filts can have a stride of 1 by which it moves one pixel at a time, or a stride of 2 and so is moving 2 pixels at a time. As bigger strides are taken, the spatial dimensions of the output are reduced resulting in smaller feature maps.

**Padding:** However, to maintain the spatial aspect of the input, padding can be added around the input image. There are two common types of padding:

* Valid Padding: There is no padding used in this case and thus the filter is convolved only over a part the image area, hence the output.
* Same Padding: A padding is done to ensure its feature map has the same spatial dimension as the input. This is most often done by placing zeros on the periphery of the reticle image.

**Activation Function:** After each convolution operation non-linearities are introduced into the model a process commonly referred to as activation. ReLu is the activation function that is widely used in CNN and is defined as f(x)=max (0, x) where all the negative values in the feature map will be set to zero while the positive values would remain the same. This non-linearity is beneficial to the network since it must learn complicated patterns with regards to the given data.

**Pooling Layer:** Pooling is another down sampling operation that solves the problem in which the spatial dimensionality of the feature maps is lowered, making the network less computationally complex and lowering the risk of overfitting. Max pooling is the most frequent type where the maximum value of a certain window (for example, 2x2) of the feature map is selected. Another type is Average Pooling where the formula used to calculate the pooled value is the average of the window. The pooling layers allow saving important attributes while not storing the less relevant information.

**Fully Connected Layer:** Finally, the feature maps are mostly made to pass through convolutional pooling and full connected layers similar to that of fully connected layers of a conventional neural network. They are charged with the responsibility of providing the final decision as to whether, for example, the image is a cat or not or the cancer is endemic in this region or not given the features provided by the convolutional layers.

**Output Layer:** The last layer of a CNN is generally an output layer which employs a SoftMax activation function (in the case of classification problems) to generate the probability distribution to the class labels.

**Concepts in CNNs**

**Dropout:** Dropout is one of the techniques used in CNNs to reduce cases of overfitting which is a common issue with the CNNs models. In training, dropout takes a random set of neurons at a given layer to be ‘dead’ or nonfunctional, hence the network will have to learn the features of data all over again. This is beneficial for the model to generalize well on the new data which it has never seen before.

**Batch Normalization:** Batch normalization is another technique that is commonly employed for stabilizing and accelerating the training process through normalization of inputs to each layer during the learning process. This way, every layer gets inputs that are uniformly distributed and this reduces the training time and makes the network to be less sensitive to initialization.

**Residual Connections:** First invoked in ResNet architectures, the identity shortcuts are pathways that bypass one or more layers and connect them directly to the later stage. Such connections reduce the effects of the vanishing gradient problem, thus enabling the training of deep networks because gradients can now pass through such a network more.

**Data Augmentation:** Thus, during the training process, data augmentation techniques are used to improve the CNNs’ resilience. These techniques include techniques that entail creating artificial workload by affording geometric transformations like resizing, rotations, flipping, zooming and translating the training data set images. This makes it easier for the model to generalize on the variations of the input data.

**Transfer Learning:** Transfer learning is a very effective approach where an already existing CNN model (that was initially trained on humongous dataset, for example ImageNet) can be retrained on a comparatively lesser set of data of a particular task. This is because by transporting the features learned from the pre-training, transfer learning drastically shrinks the training time that is necessary to achieve top notch performance especially when the target data set is small.

**Operation of CNNs in Practice**

The training of a CNN requires providing a large amount of data which has been labeled, and tweaking the weights of the filters to minimize the loss function that quantifies the dissimilarity between the network’s outputs and the actual labels. This is done using backpropagation and an optimization algorithm such as Stochastic Gradient Descent (SGD), or Adam (Mandal, 2021).

In the forward pass, the image input to the layers in the network where the convolution layers extract the hierarchical features from the input image, the pooling layers down sample the output from the convolution layers and the fully connected layers decide the output based on the features learnt in the previous layers. The loss is computed, in the process during the backward pass gradient is computed from the weights and the weights are updated in order to reduce the loss.

Time after time (epochs), the network improves in the making of the forecasts and can do well even on sets of data that has not been trained on. CNNs are most suitable when handling image data, however, they are also computationally costly approaches and hence, they may need GPUs, and large data to provide the best solution.

# Chapter 3: Methodology

## Tools and Techniques

* os Module: A command prompt is used for communication with the operating system, more specifically it is used for the managing of file and directory paths.
* cv2 (OpenCV): A library to load images or as preprocessing, post-processing and image generation tool.
* matplotlib.pyplot: A plotting library that was used in visualizing the images and the data especially when it comes to displaying sample images and the predictions that were made.
* random Module: Popular with strategies involving random sampling that is applied when choosing a random sample of images to display or in cases of shuffling of datasets.
* torch (PyTorch): That is, a tensor flow is a deep learning framework employed for construction and training of neural networks.
* torch.nn: Contains classes and modules to design and use neural networks, layers, such as `Conv2d`, `ReLU`, `Linear`.
* torch.optim: Includes optimizers such as Adam, for training the neural network by minimizing the loss function.
* torchvision.transforms: Includes regular image enhance functions like scaling, converting images to tensors, and normalization before the set can be processed.
* torch.utils.data.DataLoader: A loader tool that helps in feeding data in little portions to the model to support training and other assistances to feeding data streams into the management of the training models.
* sklearn.metrics: Scully and Zhang used a module from scikit-learn library with the same name to compute the accuracy, precision, recall, and F1 values for the model.
* train\_test\_split (from `sklearn. model\_selection`): To refer to the division where the dataset can be split into the training and testing data set so that the accuracy of the model can be attained effectively.
* torch.save and torch.load: Used for saving/storing state dictionary of the model so that the model can be resumed later or evaluated.
* torch.no\_grad(): Temporarily turns off gradient calculations to perform model evaluation for space and time efficiency.
* Data Augmentation (Image Flipping): A method to enhance the variation of the training data set by producing derivatives of the images and enhance the model resistance.
* CNN (Convolutional Neural Network): A convolutional neural network, with convolution, pooling and fully connected layers which is best used when handling image data.
* GPU Utilization: The code also determines whether there is a GPU available and if not it resorts to using a CPU to perform the computations.
* Normalization: Normalzes the pixel intensity of images to have the same mean as standard deviation and maintains consistency which aids in faster and more optimized training.

## EDA and Visualization

In the code, the number of images in folders is counted which involve the declaration of the dictionary in which the count of images in the folder is to be stored and the list of the directories in the form of subfolders in the form of path of the dataset. The images are then imported into Python as its environment as are the labels on the images.



The first action is a visualization step that entails displaying random images from each of the folders as a way of getting a feel of the dataset, which is done by using subplots. Data augmentation is done on the data by flipping the images horizontally by mirroring meaning that there is an increase in the number of training examples. The original images and the images augmented by using the above explained method are shown below. Then the images are loaded and cast into common format, reshaped to 64\*64 size, converted to tensors and normalized. To begin, the labels are translated into numbers in order to improve the training process, next, the data is divided into a training set and a testing set with proportions of 80:20. Here, a quite basic structure of the CNN model is employed where two convolutional layers with following fully connected layer are included. The model is trained over ten epochs, and loss and ETA of the model are provided epoch wise. Particularly, after the model becomes trained, the accuracy performance on the test set, precision, and recall, F-measure are computed. Another new CNN model is laid down with more convolution layers included and dropout layer for regularization is then trained in a similar manner. Similar to the previous model, evaluation of this updated model is also done based on the above-mentioned performance metrics. Still in the code, techniques on saving the model, when the GPU is available, and how to show some of the predictions are embedded. At last, the result is printed, the items consist of classification report and calculated mean value for checking the effect of both the initial model and the improved one.

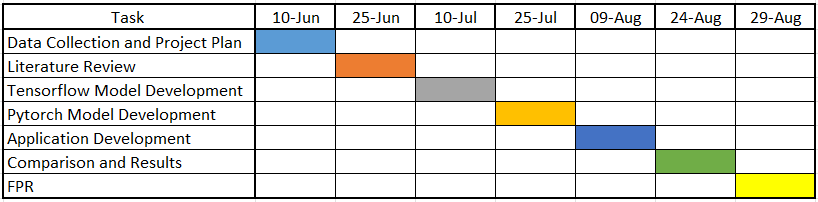


## Project Plan

The Project is divided into 7 major subsections each of which will be followed by the next. The first part of any research is to Collect the data and create a project plan that encompasses the flow and pace at which the research will be conducted. For each subsection 2 weeks of time is allotted, this is subject to change as the research progresses. If in any case one of the modules completes faster, the remaining time can be utilized to check the already completed sections and prepare for the next sections.

* TensorFlow: This framework is the most used framework to develop and train neural networks. This is a core module in this research along with the next module.
* Pytorch: This is another framework for developing neural networks that is famous for its ease of usage.

Both these main modules lay as the core to this research project. Comparison of these two modules will inherently result in development of a deep-learning based classification engine for cricket.



# Chapter 4: Results and Conclusion

## Results

Three different CNN models were applied to distinguish and analyze the efficiency of the given set of cricket shots. These models were trained and tested on four types of cricket shots: legglance-flick, pullshot, sweep, and drive are some of them. The measures obtained from these models are accuracy, precision, recall, and F1-score whereby the results were used to evaluate the proposed technique’s capability to correctly classify the shots.

**Model 1: Simple CNN**

The first employed model was a simple CNN model which had two convolution layers and a fully connected layer. While designing the PEB model, the main goal was to create a benchmark performance for the classification of cricket shots. Training the model contained several epochs while using the Adam optimizer, and cross entropy as the loss function.

**Performance Metrics**

* Accuracy: 85%
* Precision (Weighted): 86%
* Recall (Weighted): 85%
* F1-Score (Weighted): 85%

In this model the accuracy was, 85% which can also be inferred from the precision = recall = F1-score = 0. 85. The classification report showed that the model was not bad across all the four classes with an average of F1-score of 0. 83 to 0. 86 that was used in individual shots. The two shots that were defined as ‘sweep’ and ‘legglance-flick’ had slightly higher recall, which might mean that the model was capable of determining whether the shot was of this type correctly.

The Simple CNN model offered a satisfactory starting point in completing the given task. The network, albeit of a quite simple structure, succeeded in preserving some necessary characteristics in the shooting of cricket but failed to handle a ‘pullshot’ class to a significantly lower degree. This suggest that although the model was able to detect some features pretty well, it may have lost the essence of capturing other rich characteristics of some of the shots.

**Model 2: CNN with Extra Layers and Dropout**

In the second model more layers of convolution were added and some percentages of the neurons were dropped to improve the feature extractor and reduce overfitting. In this case, this model sought to increase the depth of network in the network, adding many hidden layers and dropout layers which would enhance the generalization ability.

**Performance Metrics**

* Accuracy: 91%
* Precision (Weighted): 91%
* Recall (Weighted): 91%
* F1-Score (Weighted): 91%

The chosen Updated CNN model demonstrated higher accuracy level in all of the performance indicators with 91% accuracy. The number of cases that the system marked correctly has improved the precision almost by half reaching 0. New layers and dropout mean that the author was able to skew the model in a way where it would be able to generalize convoluted with the help of more layers while dropping out the information in a way that would help it become more perceptive in the circumstances of distinguishing between training data and test data given to it, thus scoring 91. As for the quantitative results, it is crucial to note that the ‘drive’ shot class produced the highest level of recall which stood at 0. 94 increasing the model’s certainty in correctly identifying this shot.

Positive trends in measurements of performance for all the categories clearly indicate the utility of the deeper network and dropout to curb over fitting. The model was slightly more successful in differentiating between the several cricket shots due to the complex architecture of the network which enabled the algorithm to pick on subtlety present in the data-set. These dropout layers probably helped in this improvement by preventing overfitting that lowers the performance of the model on the test set.

**Model 3: CNN with five convolutional layers**

The third model proceed to the architecture to five convolutional layers. This increase in depth was supposed to let the model learn more intricate patterns in the images and, in turn, perform better.

**Performance Metrics**

* Accuracy: 89%
* Precision (Weighted): 89%
* Recall (Weighted): 89%
* F1-Score (Weighted): 89%

Notably, the performance assessment measures of this model turned out to be slightly weaker than Model 2 with an accuracy of 0. 89 and corresponding values for precision, recall, and F1-score being 0. 89. Among all the classes, the ‘pullshot’ class had the highest average recall of 0. 95, this means that the model was excellent in recognizing this specific shot. Nonetheless, there was a lesser recall of 0 among the group that featured in the ‘drive’ class. 83 meaning, therefore, that the added layers did not have a positive effect on all classes in the same manner.

The contrasting performance in Model 3 as compared to the previous model points towards the fact that the model might have become too complex in comparison to the given dataset. These extra layers could have posed a level of difficulty that could not be exploited by the model because of either overfitting or the gradient vanishing problem whereby the model finds it hard to pass gradients through the deeper layers. This highlights a critical aspect of deep learning and as in this paper, the more complex models do not necessarily imply better performance. Even the disparity in performance across different shot classes raises the same questions of potentially needing to fine-tune the architecture or training process to a greater depth for all classes.

## Comparative Analysis

It can be seen that in case of, both deeper networks as well as the one with regularization techniques like dropout, the former performs better up to a certain limit. In Model 2 architecture, the layers were made deeper and they were also regularized in some portion that helped to achieve the highest accuracy and performance. Model 1 as previously suggested is much simpler and gave a good starting point but the performances suggested the need for a better model for classification of cricket shots. Model 3 though deeper offered similar benefits and a slightly poor performance which could be consequences such as “overfitting’’ or the general problem of training a very deep learning network with the given dataset.

The classification reports also show that despite the fact that all the models were quite similar in terms of their performance, some of the classes such as ‘pullshot’ and ‘drive’ posed greater classification difficulties although these were present in all categories: simple and complex. This means that specific features in these shots are either less discernible or may need other higher-level features that these models, especially the simple one, could not extract properly.

Therefore, from the comparative analysis of the three different layers of the CNN for cricket shot classification, it becomes evident that the shallow CNN with moderated depth and liable regularization like Model 2 would perform the best among all three. As is seen in several of the classes, increasing the depth of the network helps improve the feature learning, but at a cost; it makes the network more prone to overfitting, or encountering other training issues if poorly managed. Thus, the choice of the model architecture remains critical here and should depend on the properties of the input data and the difficulty of the learning task. More enhancements might include testing with other architectural structures including those of ResNet or DenseNet, which are preconditioned for the training challenges that deep networks conduct.

## Conclusion

The results from different three CNN models indicate that the choice of an accurate architecture of the CNN and training technique for cricket shot classification can significantly affect the model’s performance. As for the simple CNN model, it was rather unsuitable for the given task due to its inability to capture the essence of the dataset, as reflected in the performance indicators. The addition of more layers and dropout in Model 2 proved to help even further showing that a deeper network with regularization can improve generalization and thus improve the ability of the model to distinguish between different cricket shots. Also, the high accuracy and high results for all metrics proved that this model was able to learn the important features of each shot type.

Yet, Model 3, which deepened the network, failed to improve the performance of the previous model, Model 2, and even slightly worsened. This implies that while increasing the model complexity improves performance there is a point of inflexion where the change in performance is negative, perhaps because over complex models are prone to overfitting or feeding forward neural networks with more than a certain number of layers on the given data may not be easy. The issues, which were identified when defining certain shot classes, ‘pullshot’ and ‘drive,’ show that in order to enhance the classification of needed samples, it either necessary to implement superior feature extraction or to fine-tune the model.

All in all, it can be stated that for better performance there is a possibility to deepen and complicate, but it is necessary to be cautious of the overfitting issue and necessity of training. According to the evaluation, the best performing model to solve this specific task was Model 2 which employs both depths and dropout.

## Future Work

The work on the classification of cricket shots applying CNNs opens a vast field of potential developments in the sphere of sports data analysis and the assisting technical means. Nevertheless, the current work can be enhanced and expanded greatly with capacities, competence, and effectiveness in the future.

Incorporation of Advanced Architectures: The newer or successive version of this project can work by establishing more complex structures of deep learning that include ResNet, DenseNet or Inception networks, considering the fact that these architectures are likely to go deep yet they do not have problems of vanishing gradients. These architectures might contribute to the improved level of details of the obtained cricket shots and therefore may help in achieving higher classification values and generalization on the unseen data.

Integration of Temporal Information: At the current time, there is a manifestation into the form of models that can classify individual frames of the cricket shots; however, temporal information could be very insightful in giving a better view of the shot. The recurrent neural networks (RNNs) or long short-term memory (LSTM) networks can be combined with CNNs to process a sequence of frames with the help of which the shot’s dynamic motion and progression, which is very important for classification, can be analyzed.

Expansion to Multimodal Data: The future work could also include the combination of other modalities of data like, audio or, sensor data in order to make our classification system more robust. For instance, one could experience audio data of the bat striking the ball and this can give further clues to differentiate between several types of hits.

Application to Real-Time Analytics: With further improvement the models could be applied in real time and the players as well as the coaches could get instant feedback during the training sessions or during a match. It would thus be very useful in professional training arrangements where practical application of the knowledge imparted is critical.

Dataset Augmentation and Diversity: An increase in the number of types of cricket shots in the dataset and increasing the size of the data will give an improved performance of the model. However, there is an opinion that including the shots taken in other conditions, players’ behaviors, and camera views could enlarge the model’s spectrum of usage in practice.

# Chapter 5: Legal, Ethical and Professional Issues

Considering several issues that are related to the usage of CNNs in cricket shot classification for the scope of this research, some legal, ethical, and professional issues are inevitably encountered in this field.

1. Intellectual Property Rights: The first of them is the violation of the law against the use of an individual’s copyrighted materials, particularly if you want to include videos from cricket matches or practice sessions. Still, if the video data used as a source for the models’ training contains copyrighted content, legal repercussions regarding the violation of copyrights arise. Special attention must be paid to the fact that all of the used video content is either public domain, properly licensed, or collected under the rights holders’ consent.

2. Fair Use and Licensing: When building a system that is used for categorizing and analyzing cricket shots there is a possibility of a breach of the fair use Act, if the system is commercialized. Particular attention must be paid to licensing of the content and legal perspectives for the definition of the fair usage especially in the context of the model trained on the TV broadcasts.

3. Player Consent and Image Rights: Another severe matter involves the invasion of privacy when the media circulates images and videos of players without their permission. In many sports, the athletes have image rights and every time the image of an athlete is used in a commercial or analytical manner it is expected to be done lawfully. The failure to get the consent might result in lawsuits and the breach of trust with professional athletes and organizations in the sporting industry.

4. Bias and Fairness: This is why in the professional sports it is very essential to make sure that the particular technology supporting the game does not favor any particular team. A model that misclassifies shots or leans toward one style of play could bring in biased results which in its turn may influence, player ranking, coaching decisions, and potentially even players’ careers. To maximize the fairness of the advantage Trading Strategy model it is crucial to thoroughly evaluate the model performance across different sets of data and do so for each of the player types and conditions.

5. Impact on Traditional Coaching Roles: The integration of automatic shot classification systems might also pose a threat to professionals amongst the conventional coaches and analyst. Utilizing such technology presents an ethical concern because it is associated with the professional opinions that have been historically used on players. There is a need to market the technology more as a supplement to coaching by people and not a substitute for it, so as to avoid competition with the traditional ways of coaching.

# References

Banoula, M. (2020) *What is TensorFlow? Deep learning libraries and program elements explained*, *Simplilearn.com*. Simplilearn. Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow (Accessed: June 8, 2024).

Bhat, I. *et al.* (2023) “Building a video dataset for cricket shot analysis,” in *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*. IEEE, pp. 1–6.

Chatterjee, C. C. (2019) *Basics of the classic CNN*, *Towards Data Science*. Available at: https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add (Accessed: August 7, 2024).

*Computer vision in sports: applications and challenges* (no date) *Superannotate.com*. Available at: https://www.superannotate.com/blog/computer-vision-in-sports (Accessed: June 8, 2024).

Fernandes, J. B. *et al.* (2023) “Cricket shot detection using 2D CNN,” in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, pp. 608–612.

Gupta, A. (2021) *A comprehensive guide on optimizers in deep learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (Accessed: August 7, 2024).

Jagadeesh, M., Rithesh and Sagar (2023) “Cricket shot detection using deep learning: A comprehensive survey,” in *2023 International Conference on Networking and Communications (ICNWC)*. IEEE, pp. 1–8.

Malviya, N. (2023) *Convolution & pooling layers - Nikita malviya*, *Medium*. Available at: https://medium.com/@nikitamalviya/convolution-pooling-f8e797898cf9 (Accessed: August 7, 2024).

Mandal, M. (2021) *Introduction to Convolutional Neural Networks (CNN)*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ (Accessed: August 7, 2024).

Mannan, M. *et al.* (2021) “Detection of cricketing activities using deep learning,” in *TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON)*. IEEE, pp. 1–6.

Radhakrishnan, P. (2017) *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?*, *Towards Data Science*. Available at: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a (Accessed: August 7, 2024).

Sarita (2023) *Basic understanding of neural network structure - Sarita, PhD*, *Medium*. Available at: https://medium.com/@sarita\_68521/basic-understanding-of-neural-network-structure-eecc8f149a23 (Accessed: August 7, 2024).

Sawtell-Rickson, J. (2022) *What is computer vision?*, *Built In*. Available at: https://builtin.com/machine-learning/computer-vision (Accessed: August 7, 2024).

Sengupta, J. (2023) *How to decide the hyperparameters in CNN - JOY SENGUPTA*, *Medium*. Available at: https://medium.com/@sengupta.joy4u/how-to-decide-the-hyperparameters-in-cnn-bfa37b608046 (Accessed: August 7, 2024).

Sharma, G. (2021) *Image classification with tensorflow: Data augmentation on streaming data (part 2)*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/05/image-classification-with-tensorflow-data-augmentation-on-streaming-data-part-2/ (Accessed: June 8, 2024).

Simplilearn (2020) *An ultimate tutorial to neural networks*, *Simplilearn.com*. Simplilearn. Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/neural-network (Accessed: August 7, 2024).

Simplilearn (2021) *What is PyTorch, and how does it work: All you need to know*, *Simplilearn.com*. Simplilearn. Available at: https://www.simplilearn.com/what-is-pytorch-article (Accessed: June 8, 2024).

Yathish, V. (2022) *Loss functions and their use in neural networks*, *Towards Data Science*. Available at: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9 (Accessed: August 7, 2024).

# Appendices

import torch

import torch.nn as nn

import torch.optim as optim

import torchvision.transforms as transforms

from torch.utils.data import DataLoader, Dataset

import os

import cv2

import time

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, classification\_report

import matplotlib.pyplot as plt

import numpy as np

# Custom Dataset class

class CricketShotDataset(Dataset):

def \_\_init\_\_(self, image\_paths, labels, transform=None):

self.image\_paths = image\_paths

self.labels = labels

self.transform = transform

def \_\_len\_\_(self):

return len(self.image\_paths)

def \_\_getitem\_\_(self, idx):

image = cv2.imread(self.image\_paths[idx])

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

label = self.labels[idx]

if self.transform:

image = self.transform(image)

return image, label

'''This Code is used to load the images into python and then convert them into RGB color format, for training and test data.

If we give the number of images we need, this code will load the required number of images'''

# Function to load images from folder

def load\_images\_from\_folder(folder\_path):

image\_paths = []

labels = []

for subdir in os.listdir(folder\_path):

subdir\_path = os.path.join(folder\_path, subdir)

if os.path.isdir(subdir\_path):

for file in os.listdir(subdir\_path):

file\_path = os.path.join(subdir\_path, file)

if file\_path.endswith(('jpg', 'jpeg', 'png')):

image\_paths.append(file\_path)

labels.append(subdir)

return image\_paths, labels

'''This code is used to load the into python similar to the first code.'''

# Define transformations

transform = transforms.Compose([

transforms.ToPILImage(),

transforms.Resize((64, 64)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),

])

'''this is a step by step functin to first convert the loaded data into image format, resize all the images to 64x64,

convert the images to a tensor,

normalize the images so that the pixel values have a consistent mean and standard deviation'''

# Load images and labels

dataset\_path = 'data' # Use the correct path to your data folder

image\_paths, labels = load\_images\_from\_folder(dataset\_path)

'''Using the above code we are loading the images into python along with their labels'''

# Convert labels to numeric values

label\_to\_idx = {label: idx for idx, label in enumerate(set(labels))}

idx\_to\_label = {idx: label for label, idx in label\_to\_idx.items()}

numeric\_labels = [label\_to\_idx[label] for label in labels]

'''Here we are converting the names of the shots into dictionaries

these dictionaries create pairs of values ex:[0:drive] this will help the algorithms to convert the english letters into numbers'''

# Split data into training and testing sets

from sklearn.model\_selection import train\_test\_split

train\_paths, test\_paths, train\_labels, test\_labels = train\_test\_split(image\_paths, numeric\_labels, test\_size=0.2, random\_state=42)

'''This code will split the data into 2 parts

first part is the training part where the algorithm will be trained

second part is used to test the algorithm that is already trained'''

# Create datasets and dataloaders

train\_dataset = CricketShotDataset(train\_paths, train\_labels, transform=transform)

test\_dataset = CricketShotDataset(test\_paths, test\_labels, transform=transform)

'''Here the images that are split into training and testing are laoded into python as datasets'''

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

test\_loader = DataLoader(test\_dataset, batch\_size=32, shuffle=False)

'''Here the datasets are converted into batches that will help the algorithm to train based on batch size'''

# Define a simple CNN model

class SimpleCNN(nn.Module):

def \_\_init\_\_(self, num\_classes):

super(SimpleCNN, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(3, 32, kernel\_size=3, stride=1, padding=1)

self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1)

self.pool = nn.MaxPool2d(kernel\_size=2, stride=2, padding=0)

self.fc1 = nn.Linear(64 \* 16 \* 16, 128)

self.fc2 = nn.Linear(128, num\_classes)

self.relu = nn.ReLU()

'''Here the Convolution Neural Network is created with different layers

after the convolution layers a normal neural network is attached for the training'''

def forward(self, x):

x = self.pool(self.relu(self.conv1(x)))

x = self.pool(self.relu(self.conv2(x)))

x = x.view(-1, 64 \* 16 \* 16)

x = self.relu(self.fc1(x))

x = self.fc2(x)

return x

''' this code is used to send the learned weights back to the algorithm to adjust it according to prediction

this will help the algorithm to learn step by step'''

# Initialize the model

num\_classes = len(set(labels))

model = SimpleCNN(num\_classes)

'''Loading the model and giving the required number of shots to be predicteds'''

# Print the model architecture

print(model)

# Move model to GPU

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model = model.to(device)

'''This code is used to check if a GPU is present or else a traditional CPU will be used'''

# Loss function and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=0.001)

'''For the algorithm to learn it has to update weights based on the error,

this optimizer will select in which direction the error is least and update the weights accordingly'''

# Training the model

num\_epochs = 10

for epoch in range(num\_epochs):

model.train()

running\_loss = 0.0

start\_time = time.time()

for i, (inputs, labels) in enumerate(train\_loader):

inputs, labels = inputs.to(device), labels.to(device)

# Zero the parameter gradients

optimizer.zero\_grad()

# Forward + backward + optimize

outputs = model(inputs)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

# Print statistics

running\_loss += loss.item()

# Calculate ETA

elapsed\_time = time.time() - start\_time

batches\_left = len(train\_loader) - (i + 1)

eta = elapsed\_time / (i + 1) \* batches\_left

print(f'Epoch [{epoch+1}/{num\_epochs}], Step [{i+1}/{len(train\_loader)}], Loss: {loss.item():.4f}, ETA: {eta:.2f} seconds')

print(f'Epoch [{epoch+1}/{num\_epochs}], Loss: {running\_loss/len(train\_loader):.4f}')

print('Finished Training')

'''The code defines a training loop for a CNN model over ten epochs.

For each epoch, it sets the model to training mode and initializes the running loss and start time.

It iterates over the training DataLoader, transferring inputs and labels to the device (GPU/CPU).

Gradients are zeroed before each batch to prevent accumulation.

The model makes predictions, calculates the loss using Cross-Entropy Loss, and updates the model parameters via backpropagation

using the Adam optimizer.

Running loss is accumulated, and the estimated time remaining (ETA) is calculated and displayed for each batch.

At the end of each epoch, the average loss is printed, providing a performance summary.'''

# Save the model

torch.save(model.state\_dict(), 'simple\_cnn\_cricket\_shot\_model.pth')

'''The model is saved in a dictionary format'''

# Evaluation on the test set

model.eval()

y\_true = []

y\_pred = []

with torch.no\_grad():

for inputs, labels in test\_loader:

inputs, labels = inputs.to(device), labels.to(device)

outputs = model(inputs)

\_, preds = torch.max(outputs, 1)

y\_true.extend(labels.cpu().numpy())

y\_pred.extend(preds.cpu().numpy())

'''The code evaluates the trained CNN model on the test set.

It sets the model to evaluation mode and initializes lists for true and predicted labels.

Within a `torch.no\_grad()` block, it iterates over the test DataLoader, moving inputs and labels to the device.

The model makes predictions, and the predicted class with the highest score is selected.

These predictions and true labels are collected into lists. T

his process ensures no gradients are calculated, making it memory efficient and faster for evaluation.'''

# Calculate performance metrics

accuracy = accuracy\_score(y\_true, y\_pred)

precision = precision\_score(y\_true, y\_pred, average='weighted')

recall = recall\_score(y\_true, y\_pred, average='weighted')

f1 = f1\_score(y\_true, y\_pred, average='weighted')

report = classification\_report(y\_true, y\_pred, target\_names=[idx\_to\_label[i] for i in range(num\_classes)])

'''Here all the required performance metrics are calculated for us to check the model performance'''

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1 Score: {f1:.4f}')

print('\nClassification Report:\n', report)

'''The calculated metrics are printed'''

# Plotting some predictions

fig, axes = plt.subplots(3, 3, figsize=(12, 12))

axes = axes.flatten()

for i in range(9):

idx = random.randint(0, len(test\_paths) - 1)

img = cv2.imread(test\_paths[idx])

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

true\_label = idx\_to\_label[test\_labels[idx]]

pred\_label = idx\_to\_label[y\_pred[idx]]

axes[i].imshow(img\_rgb)

axes[i].set\_title(f'True: {true\_label}\nPred: {pred\_label}')

axes[i].axis('off')

plt.tight\_layout()

plt.show()